Université de Paris

PRESENTATION PROJET PUISSANCE 4

Par

David Halioua & Emmanuel Elbaz

Langage De Programmation : C

Professeur : Mikal Ziane

Emmanuel :

1) struct Tableau2D { unsigned int nbColonnes; unsigned int nbLignes; char *cases; };
2) struct joueur{ char nom[100]; int couleur; };
3) enum occupation { vide,rouge,jaune };
4) int xy2i (int ligne, int colonne) ;
5) void init(Tableau2D*, unsigned int nbColonnes,unsigned int nbLignes);
6) void afficher (Tableau2D* t);

David :

1) Int placerGrille(Tableau2D* t,int x,int couleur);
2) int quatre_diago1N(Tableau2D* t,int x,int y,int couleur) ;
3) int quatre_diago2N(Tableau2D* t,int x,int y,int couleur);
4) int ligneN(Tableau2D* t,int x,int y,int couleur);
5) int colonneN(Tableau2D* t,int x,int y,int couleur);
6) int gagneN(Tableau2D* t,int couleur);

PS : Toutes les fonctions test du programme sont déclarer en // dans « le main » pour raison de clarté .

Il faudra donc supprimer les \\ pour pouvoir tester le programme.

Université de Paris

Présentation du programme principale (main) :

- Le projet puissance 4 a été conçu avec le langage c

Il comprend un tableau2D qui as une structure composée de 7 colonnes et 6 lignes et d'un champ case qui est lui-même de 42 caractères qui représentent les 3 valeurs possible de la grille de jeu

(vide, Rouge(x), Jaune(O))

- Le premier joueur (j1 ou j2) est choisi au hasard grâce à la fonction « rand »
  La fonction « srand » permet d'initialiser le générateur de nombres aléatoires (la fonction rand)
  On as donc 2 valeur 0 ou 1 ,en divisant le reste par 2.
  On tire une valeur parmi les 2 joueurs
  on va permuter les joueurs dans le if que si commence =1
  c'est-à-dire que si on ne rentre pas dans le if c'est le joueur j1 qui commence la partie (commence =0),
  sinon le joueur j2 commence la partie (commence =1)

- Affectation des joueurs : j1 couleur rouge ,j2 couleur jaune
- Initialisation de la grille:
  Par définition pjoueur = j1 , padversaire=j2

  Définition du nombre de ligne et de colonne

  toutes les cases = vide

  Initialisation du compteur des 42 cases de partie nul

- Test de la fonction assertTrue et assertFalse :
  On définit ''cond'' de type booléen comme égale a « true »
  Et on teste donc notre fonction :
  On obtient test réussit pour asserttrue (car cond = true)
  et test échouée pour assertfalse (car cond = false)

- Jeu :
  On est dans le jeu tant que la condition fin est égale à 1
  C'est-à-dire tant qu'aucun joueur n'a pas encore gagner on reste dans la boucle

  Un premier test élimine les erreurs faites par les caractères entrés par le joueur
  (Ex : si il rentre un caractère au lieu d'un chiffre)
  le programme s'arrête pour empêcher le joueur de rentrer des fausses valeurs au programme et ainsi sécuriser la fonction « scanf » qui récupère les données entrées par le joueur

Université de Paris

- Placer Grille :

  Cette fonction fait descendre les jetons dans les colonnes , dans les cases disponibles

  Si notre compteur (cte) décompte 42 cases remplit de la grille, il déclare alors ''match nul'' et fin de partie (égale a 0) .

  Si le joueur gagne en alignant 4 cases (horizontale, verticale, diagonale), il déclare alors ''Bravo joueur '' la partie est gagner par le joueur et fin de partie(égale a 0) .

  On passe par la suite à l'autre joueur dès que le 1er joueur a rempli sa case

  C'est-à-dire qu'on fait une permutation circulaire de chaque joueur

  ''Ptemp'' prend de la valeur de ''Pjoueur''

  ''Pjoueur'' prend la valeur de ''Padversaire''

  ''Padversaire'' récupère alors la valeur de ''Ptemp''

- En fin de jeu on libère la mémoire alloue grâce à la fonction « malloc »

COLONNES :

Le programme qui définit un numéro de colonne entre 1 et 7 , qui est nommé x .

LIGNES :

Le programme qui définit un numéro de ligne entre 1 et 6 , qui est nommé y.

Il y a 6 lignes dans le jeu

Au lancement du jeu le joueur commence a la ligne 0

xy2i :

Cette fonction convertit la case y,x en l'indice du tableau 1d .

Les 42 cases de la grille sont stocker dans un tableau de longueur 42

xy2i fait la correspondance entre le tableau et la grille

X : numéro de colonnes rentrer par le joueur de 1 a 7

Y : numéro de lignes calculer par le programme de 1 a 6

Université de Paris

xy2i(y,x) = 7*(y-1)+(x-1)

le tableau va de 0 à 41 , on calcule le 1ere indice du tableau correspondant au décalage

Remplissage de la grille avec la fonction xy2i représente l'indice du tableau de taille 42:

0 **(1,1)**	1**(1,2)**	2**(1,3)**	3**(1,4)**	4**(1,5)**	5**(1,6)**	6**(1 ,7)**
7**(2,1)**	8	9	10	11	12	13
14 **(3,1)**	15	16	17	18	19	20
21 **(4,1)**	22	23**(4,3)**	24	25	26	27
28 **(5,1)**	29	30	31	32	33	34
35 **(6,1)**	36 **(6,2)**	37 **(6,3)**	38**(6 ,4)**	39**(6,5)**	40**(6,6)**	41**(6,7)**

Tableau 2D xy2i(y,x) : en bleu les coordonnées (ligne,colonne)

Partie Emmanuel

1) **struct Tableau2D { unsigned int nbColonnes; unsigned int nbLignes; char *cases; };**

   la structure du tableau2d comprend 3 éléments :
   - unsigned int nbColonnes : la variable colonnes entier positif du tableau 2d
   - unsigned int nbLignes : la variable lignes entier positif du tableau 2d
   - char *cases : la chaine de caractère cases qui est le tableau de 42 valeurs de la grille

2) **struct joueur{ char nom[100]; int couleur; };**
   Joueur (j1/j2) = couleur

   La structure joueur comprend les variables :

   -char nom[100] : le tableau nom comprend 100 caractères rattachés aux nom des joueurs

   -int couleur : la variable couleur entier positif correspond aux numéros des joueurs j1,j2

3) **enum occupation { vide, rouge , jaune };**

   Ce mot clefs énumération attribue les noms : vide , rouge(x) et jaune (0) aux différentes occupations du tableau 2d (0,1 ,2)

4) **int xy2i (int ligne, int colonne) ;**
   La fonction xy2i de type nombres entier convertit la case repérer par sa ligne(y) et sa colonnes(x) en l'indice du tableaux ''i''.
   Le return renvoie la formule mathématique : 7*(ligne -1) +(colonne -1)

Université de Paris

(reprise du cours)

7*(y-1)+(x-1)
Cela retourne l'indice (l'adresse) du tableau en fonction de la position x,y du point.

5) **void init(Tableau2D* t, unsigned int nbColonnes,unsigned int nbLignes);**
Données entrées :

La fonction tableau2D prend comme argument
- t qui est de type tableau 2d
-nombres colonnes :
-nombres de lignes :
Elle affecte :
1) Le nombre de colonnes au tableau 2d
2) Le nombre de lignes au tableau 2d
3) Elle alloue la mémoire des 42 termes du tableau case 1d
4) Elle remplit les 42 valeurs du tableau par «vide »

6) **void afficher (Tableau2D* t);**

Elle affiche la grille du jeu en affichant les valeurs du tableau t->cases[i] de dimension 42

Details :

La fonction parcourt les 42 valeurs cases du tableau :

Si la case est « vide » on affiche à l'écran le caractère ''.''

Si la case est rouge on affiche à l'écran le caractère ''X''

Si la case est jaune on affiche à l'écran le caractère ''O''

void testInit(Tableau2D* t) :

La fonction « testinit » affiche la grille du jeu en affichant les valeurs du tableau de dimension 42 .
La fonction test consiste pour les :
-nombres de colonnes : Tester les valeurs pour un nombre de colonnes égale à 7.
-nombres de lignes : Tester les valeurs pour un nombre de lignes égale à 6.
-cases de la grille :
Elle parcourt les cases de la grille sur les 42 valeurs du tableau 2D
Et elle test les valeurs de la case :
- Vide = 0
- Rouge = 1
- Jaune = 2

Université de Paris

Partie DAVID

7) **Int placerGrille(Tableau2D* t,int x,int couleur);**

La fonction grille place le jeton du joueur dans la grille et dans la colonne choisit par le joueur
Pour ce faire elle cherche la case la plus basse dans la grille soit (6,1)
Et pour ce faire elle incrémente la position y la plus basse disponible (vide)
A quoi sert les return y

testPlacerGrille(Tableau2D* t,int x)
rappel x = COLONNES définit par le joueur
La fonction PlacerGrille retourne le numéro de la ligne ou le jeton est positionner par le programme .
Le test consiste à tester parmi les 6 lignes , s'il y a bien un seul jeton trouver par le programme.

8) **int quatre_diago1N(Tableau2D* t,int x,int y,int couleur) :**
Cette fonction vérifie s'il y a quatre jetons alignés en diagonal. La diagonal est celle d'en bas à gauche vers en bas à droite.
Si les quatre cases alignées ont la même couleur (relatif au joueur en question), on retourne la valeur 0. Dans le cas contraire c'est la valeur 1 qui est retourné.
Ces valeurs seront utilisées dans la fonction gagneN.

Fonction test : On fait aligné quatre jeton d'un même joueur et on vérifie si la valeur envoyé est la bonne.

9) **int quatre_diago2N(Tableau2D* t,int x,int y,int couleur);**

Comme pour la fonction diagonale précédente, cette fonction permet de trouver l'autre diagonal du tableau qui se lit d'en haut à gauche à en bas à droite. Le test est le même que pour l'autre fonction de diagonal.

10) **int ligneN(Tableau2D* t,int x,int y,int couleur) :**

Cette fonction permet de repérer un alignement horizontal, cad sur une même ligne.
On vérifie la fonction en voyant si la fonction renvoie bien un 0 pour 4 jetons alignés.

Université de Paris

11) **int colonneN(Tableau2D* t,int x,int y,int couleur)**

Cette fonction détermine si 4 jetons sont alignés à la verticale. Elle cherche donc un alignement de quatre jetons sur une même colonne. Elle renvoi un 0 ou 1 et on la vérifie de la même manière que pour les 3 fonctions précédentes.

12) **int gagneN(Tableau2D* t,int couleur);**

Cette fonction permet de mettre fin au jeu si jamais il y a alignement de quatre jetons.
Si les fonctions appelées renvoient des 0, le programme s'arrêtera dans le main et indiquera le vainqueur. Test identique que pour les dernières fonctions.